EXECUTIVE SUMMARY

A MICROPROCESSOR BASED HIGH SPEED PACKET SWITCH FOR SATELLITE COMMUNICATIONS

GRANT NO.

NSG3191

with

National Aeronautics and Space Administration
LEWIS RESEARCH CENTER
CLEVELAND, OHIO

James Rotnem - Project Officer

Mohammed Arozullah - Principal Investigator

Stephen C. Crist - Co-Investigator

Grant Title: Design of a Microprocess-Based High Speed Space Borne Message Switch

April 15, 1978 - May 30, 1980

CLARKSON COLLEGE OF TECHNOLOGY
POTSDAM, NY 13676

ABSTRACT

This report is concerned with the design and evaluation of a microprocessor based high speed space-borne packet switch. Three designs, namely a single, three and multiple processor designs, are presented. System architectures for these three designs are presented. Further, the hardware circuits, and software routines required for implementation of the three and multiple processor designs are also presented. A bit-slice microprocessor is used. This processor has been designed and microprogrammed. Maximum throughput has been calculated for all three designs. Queue theoretic models for these three designs have been developed and utilized to obtain analytical expressions for the average waiting times, overall average response times and average queue sizes. From these expressions graphs have been obtained showing the effect on the system performance of a number of design parameters.

TABLE OF CONTENTS

1. INTRODUCTION

The purpose of the research supported under this grant was to evaluate the feasibility of using microprocessors to control satellite-borne packet switching. This was accomplished by designing a packet switch architecture suitable for microprocessor control, designing the processor(s) using 2900 series components, and evaluating the packet switch in terms of system throughput, delay, and queue sizes within the packet switch.

This work assumes that the packet switch has N serial bit streams in packet form, where N is the number of users. Output from the switch is accomplished by loading packet-sized buffers and transferring control to external circuitry to transfer data to the channel. There is one output buffer per user.

The first packet switch architecture studied uses a single microprocessor. In doing this design, it was found that three distinct software routines were needed. Under normal circumstances, each packet passing through the switch must be serviced by each of the three routines. The second design utilizes three distinct processors, one for each of the software functions. In the final design an attempt was made to use multiple processors within each of the three functions. However, it was found that to minimize contention among processors, one of the functions must be divided into two functions. It is then possible to use several processors for each of the four functions.

Some consideration was given to protocols that could be supported by the packet swtich. It was found that a full ARQ

protocol could not be used because of on-board storage requirements on the packet switch. Furthermore, it was found that any protocol not transparent to the switch resulted in significant performance degradation due to the increased processing required. For this reason, no serious consideration of protocols was made with respect to the latter two designs.

The header of each packet is protected by an error-correcting code for any protocol because, as a minimum, the packet switch must correctly identify the destination of a packet. The on-board decoding can generally be accomplished by a ROM look-up table because of the limited length of the header. Any other error correction is done by the receiving ground station.

A single fixed packet length is required by the packet switch. Any other scheme results in a significant increase in both hardware and software.

2. SYSTEM ARCHITECTURES

Because throughput is of critical importance, sufficient hardware is provided such that the processor(s) is the limiting item in system performance. This premise leads to the following properties of the proposed system architectures.

1) All data transfers are done serially. This eliminates the need for processor storage of data.

2) All internal serial transfers are performed and terminated by hardware. This allows the processor(s) to initiate a transfer and move on to the next task.

3) The processor is not required to access the data portion of the packets, but only the header.

The system architecture for the one and three processor designs are essentially identical. The single processor version is shown in Figure 2.1, and the three processor version is shown in Figure 2.2. The operation of both switches is explained by following a packet through the system.

The routing of the users' messages begins with the buffering of all incoming packets. Each input line is double buffered. Even with double buffering, the processor service response time must be short. Buffer overflow will destroy packets left too long in a buffer. In order to avoid packet losses, a minimum of processing is done at the input buffers. As soon as a full buffer is detected, the processor immediately stores the packet in temporary storage. This storage area is constructed of shift registers arranged in an array.

Once stored in the shift register array, each packet receives additional service. Their headers are decoded by the processor to determine each packet's destination. The routed packets are assigned to software output queues. Use of software queues eliminates the need for additional packet transfers required by hardware queues. Each queue corresponds to one unique output buffer.

When an output buffer becomes empty, the processor accesses the associated queue for the next packet awaiting transmission.

Each queue contains the location of each routed packet in the array awaiting transmission to that queue's corresponding output buffer. Using this information, the processor begins the transfer of the queue's oldest packet to the proper buffer. Once in the buffer, the packet is then transmitted onto the network channel under hardware control.

The software required to control the packet switch consists of three routines: The input service routine, the routing service routine and the output service routine.

In designing the multi-processor version, careful consideration was given to eliminating resource contention, because this would cause the processors to wait for hardware. One of the more significant changes from the first two designs is that the routing function is divided into two functions, sorting and routing. This was necessary because otherwise all routing processors would in general have to access all output queue lists. The function of the sorting processors is primarily to assign the packet to a specific routing processor, depending on the destination. Thus, each output queue list is accessed by one routing processor and one output processor. The general configuration of the multiple processor packet switch is shown in Figure 2.3.

The first function of the switch is to receive and to store each incoming packet. When a packet arrives, it is temporarily stored in an input buffer. An input buffer containing a newly

received packet requests processor service. Dedicated hardware pollers sequentially scan their assigned group of input buffers searching for full buffers. One group of input buffers is assigned to one Input Processor. Upon finding a full buffer, a polling circuit signals the Input Processor it is serving. Immediately, this processor establishes a data link between the full buffer and the Shift Register Array. In order to set up this link, the processor must first find an available data path in the processor's dedicated Input Switching Network. Next, the processor must find an empty location in the Shift Register Array. Once the address of an empty location is fetched from the Empty Shift Register List (ELIST), the processor completes the data link. The processor then initiates the packet's serial transfer into the array. As in the previous systems, this transfer is hardware monitored and terminated, allowing the processor to move on to a new task.

The second function of the switch is to sort each packet in the array into groups of packets that are destined for the same group of ground stations. Each unique group of stations is serviced by one unique Routing Processor. Shift registers containing newly arrived packets signal for Packet Sorting Processor service. Dedicated hardware pollers scan their assigned group of shift registers for new packets. Once a polling circuit locates a new packet, the Sorting Processor it is serving is notified. This processor fetches the packet's header and corrects it. The packet's destination is then read

from the header. Using this information, the Sorting Processor sends the packet's destination information and array address to an input/output port associated with the packet's destination. Each different I/O port belongs to one unique Packet Routing Processor. Any Sorting Processor may access any I/O port.

The Packet Routing Processors carry out the switch's third function, which is the updating of the Output Queue Lists with the addresses of sorted packets. Once an I/O port is found to contain valid packet routing data, the I/O port polling circuit signals the Routing Processor it serves. The Routing Processor responds by fetching the packet's destination information. Using this information, the processor determines to which ground station the packet is destined. Packets leave for a ground station via an output buffer which corresponds to that ground station. Each output buffer is assigned to only a single ground station. In order to route a packet to a particular ground station, the Routing Processor must assign the packet to the software output queue list which corresponds to the proper output buffer. This assignment is made by fetching the packet's array address from the I/O port and placing it into the proper queue list. Each Routing processor controls a unique group of output queue lists. A packet is considered routed once its array address is placed into one of the N queue lists.

The fourth and final function of the switch is to transmit the routed packets to their final destinations. This job belongs to the Output Processors. When an output buffer empties

due to a completed packet transmission, the buffer requests processor service. Dedicated hardware pollers sequentially scan their own group of output buffers in search of empty buffers. When an empty buffer is found by a polling circuit, the Output Processor served by this poller is informed. The processor then accesses the output queue list belonging to the empty buffer. The address of the oldest packet waiting for transmission to this destination is fetched from the queue list. Next, the processor finds a free data path in its dedicated Output Switching Network. A link is established between the shift register containing the packet to be transferred and the empty buffer via the free data path. Once this link is complete, the packet transfer is initiated by the processor. Automatic hardware controls this serial packet transfer. As soon as an output buffer is loaded, the packet is automatically transmitted to the ground station by hardware external to the packet switch. While the internal hardware transfer takes place, the Output Processor updates ELIST by placing the packet's array address into ELIST.

3. PERFORMANCE EVALUATION

Performance of the proposed designs were evaluated in terms of maximum throughput, average waiting times, overall average response times and average queue sizes.

In computing the throughputs of the various architectures it should be noted that the processing times are the determining

factor. This processing time, in all cases, is not a function of the packet size. (The delay encountered by a packet in the system does depend on packet size). The results of the throughput calculations are in terms of packets/sec.

The one and three processor architectures have an upper bound of approximately 150,000 and 500,000 packets/sec. respectively. These numbers correspond to a utilization factor of one, and thus are not realizable without theoretically infinite delay and queue sizes. The throughput of the multiple processor design depends on the number of processors. A fundamental limitation of 10^7 packets/sec. exists, but if 10^4 bits/packet are permitted this results in a system throughput well above projected needs. As an example of the multiple processor capability, it was found that 21 processors are required for an upper bound of 3×10^6 packets/sec. provided the number of users is at least 7. This corresponds to a throughput of 30 Gbit/sec. for a 10K packet length.

Queueing analyses were performed for all three systems. It was generally found that the delay times were considerably less than the propagation delay and the internal queue sizes were reasonable for utilization factors less than .7. Graphs showing the effect of the various design parameters on the average waiting times, overall average response times and average queue sizes for the three proposed designs have been obtained and are found in the final report.

4. CONCLUSION

The most significant result of this research is that system throughputs sufficient for applications currently being considered are obtainable with a processor-controlled packet switch. The delay times and queue sizes for such a switch are reasonable. The technology required for the switch is currently available and is military qualified. The production cost of the single processor packet switch is estimated at $500,000. (This does not include cost of development). The three processor version, which can support a maximum throughput of 500,000 packets/sec. is estimated to have a production cost of under $1,000,000. For the multiple processor system, cost is believed to be proportional to throughput, with $1,000,000 for 500,000 packets/sec. being the proportionality factor. It should be noted that the design and development of efforts required to get any of the packet switches proposed into the production phase is significant.

5. PUBLICATIONS RESULTING FROM THE GRANT

The following publications have resulted from the work sponsored by NASA under this grant.

1. J.F. Burnell, S.C. Crist and M. Arozullah, "Architecture and Evaluation of a Packet Switch for Satellite Applications", Conference Record of the IEEE International Conference on Communications, June, 1979.

2. S.C. Crist, J.F. Burnell and M. Arozullah, "A Microprocessor Based Space Borne Packet Switch", Proceedings of 1979 IEEE National Telecommunications Conference, November, 1979.

3. M. Arozullah, S.C. Crist and J.F. Burnell, "A Microprocessor-Based High Speed Space-Borne Packet Switch", IEEE Transactions on Communications, COM-28, 1, pp. 7-21, January, 1980.

4. J.F. Burnell, S.C. Crist and M. Arozullah, "Microprocessor Utilization in Satellite-Borne Packet Switching", jointly in IEEE Transactions on Computers, C-29, 2, pp. 206-208, February, 1980, and IEEE Journal of Solid State Circuits, SC-15, 1, pp. 142-144, February, 1980.

5. P.N. Jean, S.C. Crist and M. Arozullah, "Multi-Microprocessor Based Architecture for a Space Borne Packet Switch", Proceedings of COMPCON, February, 1980.

6. M. Arozullah, P.N. Jean and S.C. Crist, "Evaluation of a Three-Processor Architecture for a Satellite Packet Switch", Conference Record of the IEEE International Conference on Communications, June, 1980.

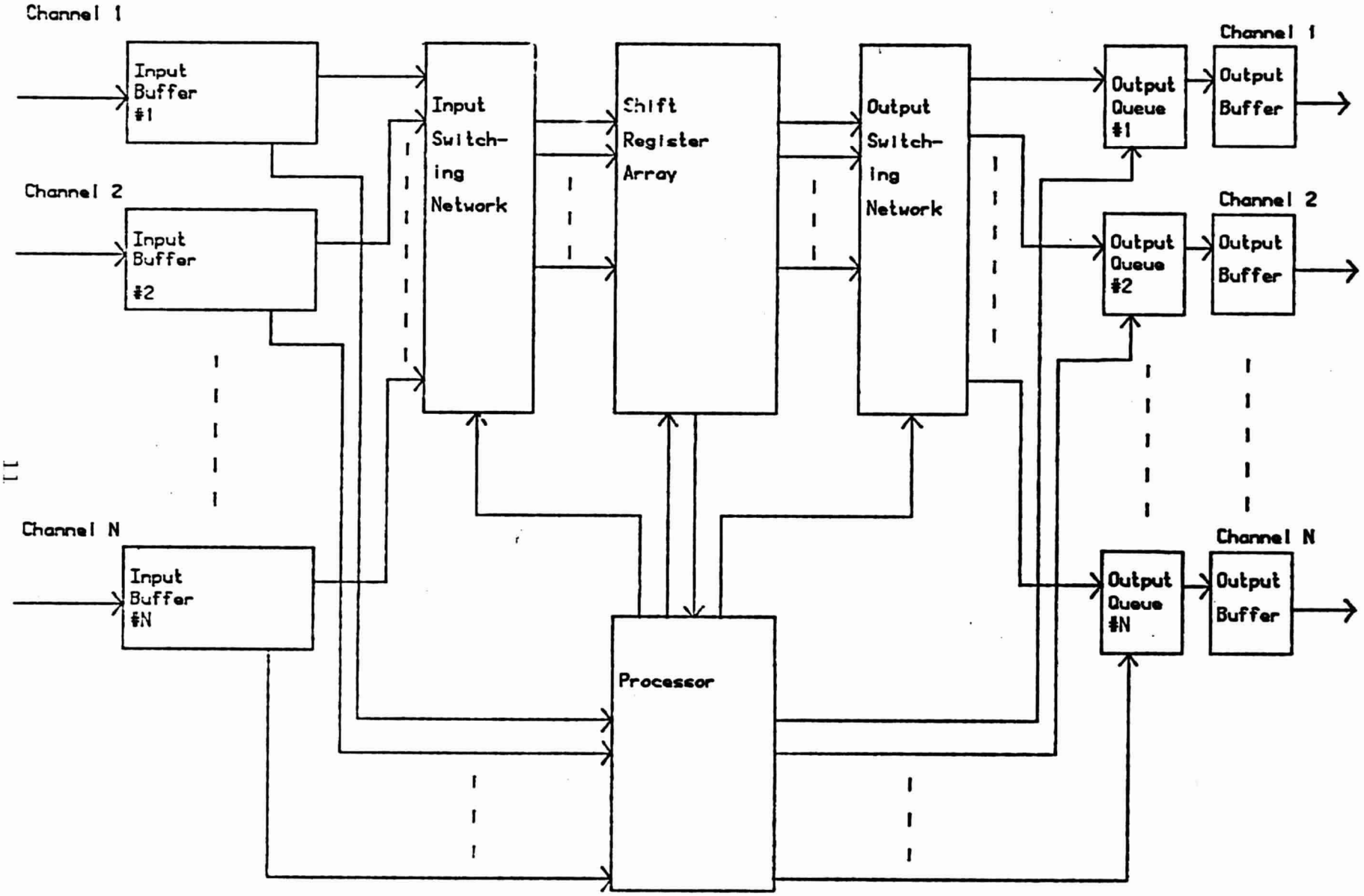

Figure 2.1. Basic Packet Switch Architecture

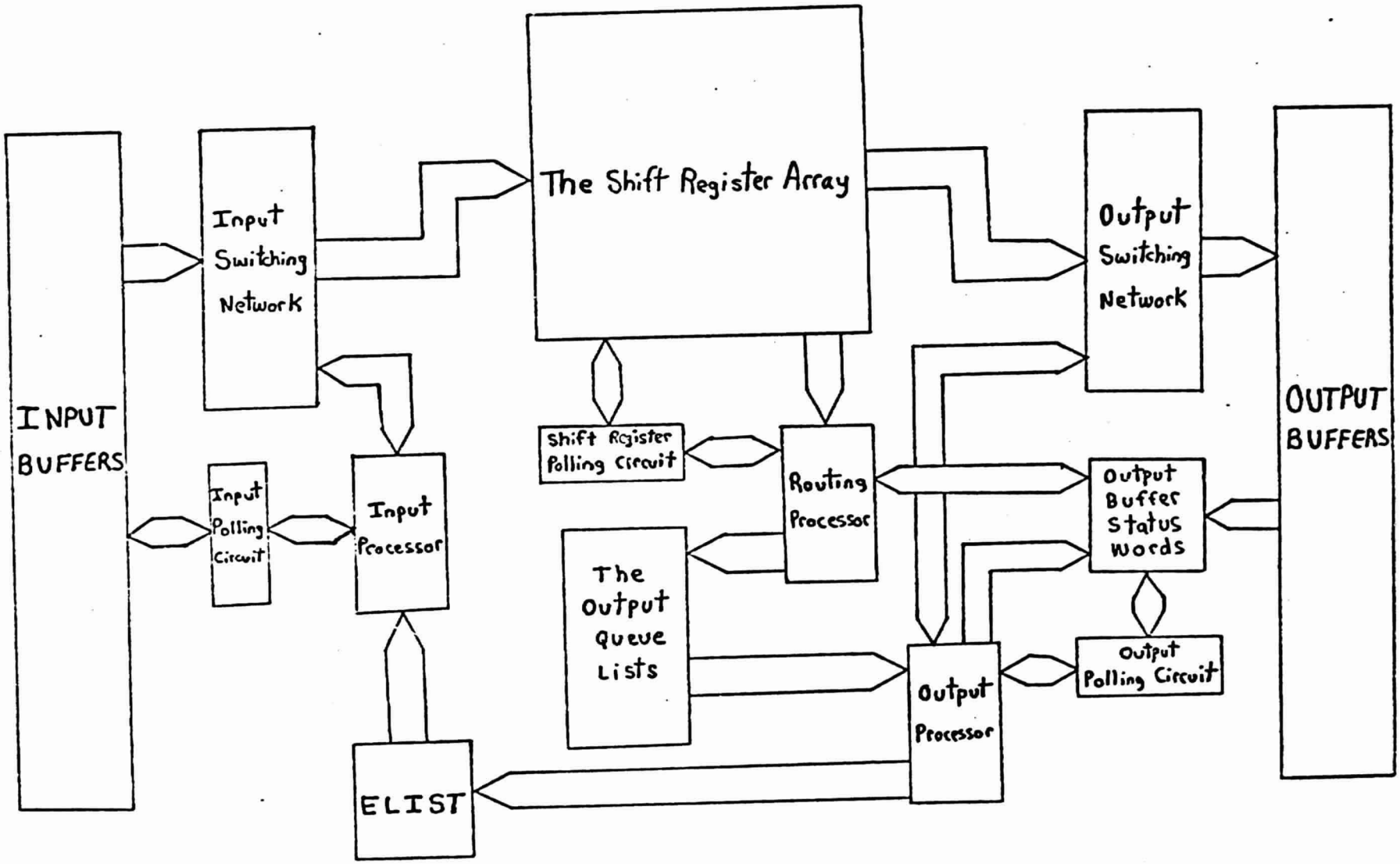

Figure 2.2. The Three Processor Architecture

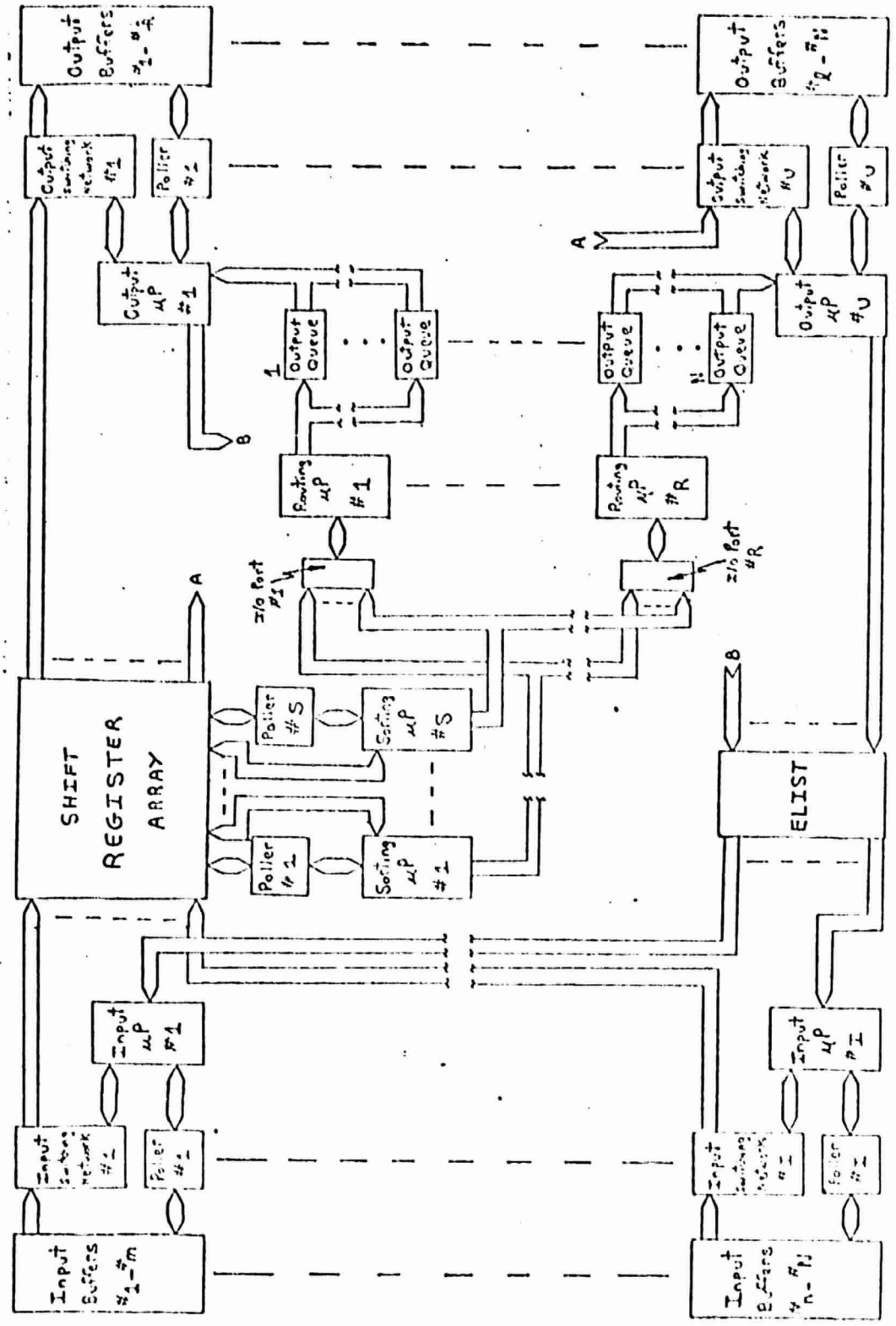

Figure 2.3. Multiple Processor Architecture